LAMPIRAN

1. Kriteria Kelulusan, Catatan Pendadaran dan Keputusan Hasil Ujian

PEMBERITAHUAN SEBELUM UJIAN :
Pengumpulan akhir dokumen Tugas Akhir/Skripsi melewati batas akhir ganjil 2022/2023, mahasiswa harus menyelesaikan registrasi dan KRS semester berikutnya.

KRITERIA KELULUSAN UJIAN SIDANG / PENDADARAN

1. Lulus ujian tanpa syarat, disebut kriteria 1.

2. Lulus bersyarat, disebut kriteria 2, yaitu dengan sedikit perbaikan atau penyempurnaan text dan atau program dalam waktu maks sampai tanggal 30 April 2023 dan tidak ada ujian lagi. Jika dalam waktu yang ditentukan mahasiswa tersebut tidak dapat menyelesaikan, maka, mahasiswa yang bersangkutan dianggap tidak lulus ujian.

3. Tidak lulus ujian sidang/pendadaran, disebut kriteria 3, dijelaskan, disarankan Ketua Tim Penguji untuk mempelajari ulang materi, merombak program/teks, atau mengganti judul.

Ketentuan bagi peserta yang tidak lulus ujian sidang / pendadaran.

1) Mahasiswa wajib menempuh ujian sidang/pendadaran ulang

2) Kesempatan ujian sidang/pendadaran ulang hanya diberikan dalam rentang waktu maksimum 6 bulan, setelah ujian sidang/pendadaran

3) Jika sampai batas waktu maksimum 6 bulan tersebut belum dapat diajukan/diselesaikan, maka calon peserta ujian dinyatakan sebagai mahasiswa peserta Skripsi/TGA baru, dengan segala ketentuan yang berlaku bagi peserta baru

4) Mahasiswa yang akan menempuh ujian sidang/pendadaran ulang ini diwajibkan membayar biaya ujian setara 2 SKS praktik, sesuai tahun angkatan

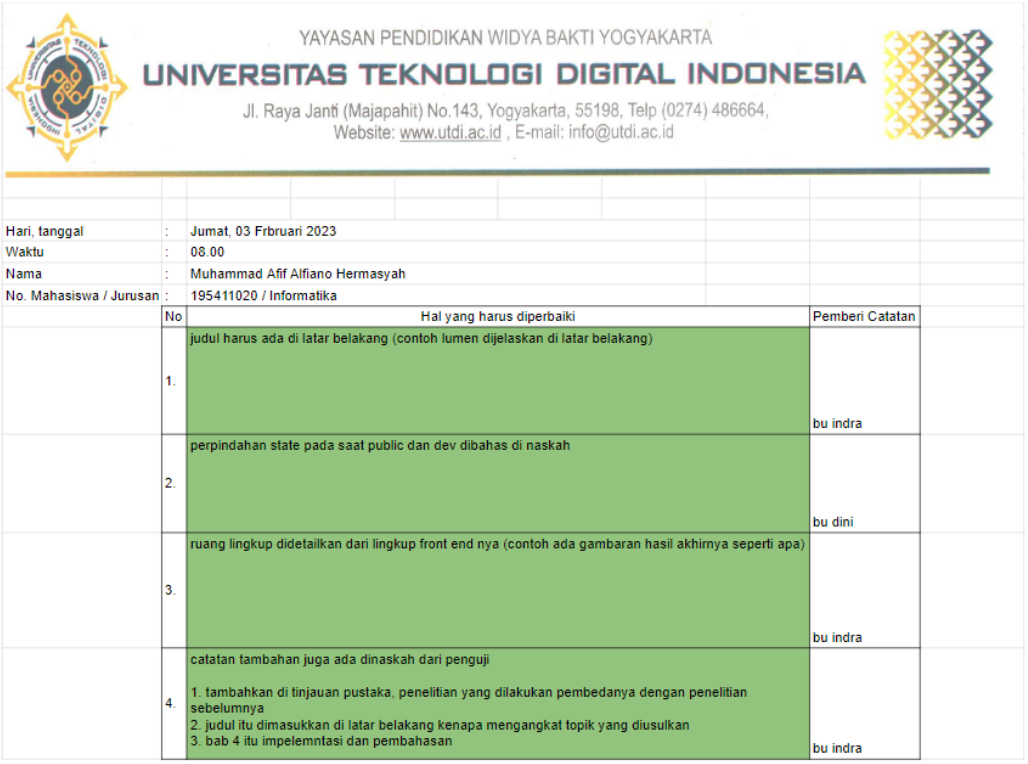
Yogyakarta, _____

Memahami dan bersedia

Mematuhi peraturan di atas,

Nama Mahasiswa

Lampiran 1 Kriteria Kelulusan



YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

Hari, tanggal	:	Jumat, 03 Frbruari 2023
Waktu	:	08.00
Nama	:	Muhammad Afif Alfiano Hermasyah
No. Mahasiswa / Jurusan	:	195411020 / Informatika

No	Hal yang harus diperbaiki	Pemberi Catatan
1.	judul harus ada di latar belakang (contoh lumen dijelaskan di latar belakang)	bu indra
2.	perpindahan state pada saat public dan dev dibahas di naskah	bu dini
3.	ruang lingkup didetailkan dari lingkup front end nya (contoh ada gambaran hasil akhirnya seperti apa)	bu indra
4.	catatan tambahan juga ada dinaskah dari penguji 1. tambahkan di tinjauan pustaka, penelitian yang dilakukan pembedanya dengan penelitian sebelumnya 2. judul itu dimasukkan di latar belakang kenapa mengangkat topik yang diusulkan 3. bab 4 itu impelemntasi dan pembahasan	bu indra

Lampiran 2 Catatan Pendadaran

YAYASAN PENDIDIKAN WIDYA BAKTI YOGYAKARTA

UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA

Jl. Raya Janti (Majapahit) No.143, Yogyakarta, 55198, Telp (0274) 486664,
Website: www.utdi.ac.id , E-mail: info@utdi.ac.id

KEPUTUSAN HASIL UJIAN PENDADARAN

Sesuai dengan hasil sidang pendadaran pada tanggal maka

Nama Mahasiswa	Muhammad Afif Alfiano Hermasyah
NIM / Program Studi	195411020 / Informatika
Jenjang	S1

dinyatakan Jabatan dengan kriteria

Ketua Penguji Indra Yatini Buryadi, S.Kom., M.Kom.

Lampiran 3 Keputusan Hasil Ujian

2. Surat Keterangan Persetujuan Publikasi Pada Eprints

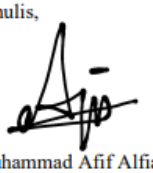
SURAT KETERANGAN
PERSETUJUAN PUBLIKASI

Bahwa yang bertanda tangan dibawah ini:

Nama : Muhammad Afif Alfiano Hermasyah
NIM : 195411020
Program Studi : Informatika
Email : muhammad.afif@students.utdi.ac.id
Judul Skripsi : Implementasi *Front-End* Aplikasi Keluhan Pelanggan PT. Jembatan Citra Nusantara Berbasis Web Menggunakan *React Javascript* dan *Redux*

Menyerahkan karya ilmiah kepada pihak perpustakaan UTDI dan menyetujui untuk **diunggah ke Repository** Perpustakaan UTDI sesuai dengan ketentuan yang berlaku untuk kepentingan riset dan pendidikan.

Yogyakarta, 18 Februari 2023
Penulis,



Muhammad Afif Alfiano Hermasyah
NIM. 195411020

Lampiran 4 Surat Keterangan Persetujuan Publikasi Pada Eprints

3. Listing Program

Komponen App.js untuk routing dan render komponen

```
import React from 'react';
import { Route, Routes } from 'react-router-dom';
import './App.css';
import { Toaster } from 'react-hot-toast';
import loadable from '@loadable/component'

// auth
import SignIn from './pages/auth/sign_in/SignIn';
import SignUp from './pages/auth/sign_up/SignUp';
import VerificationEmail from
'./pages/auth/verification_email/VerificationEmail';
```

```
import VerificationSuccess from
'./pages/auth/verification_success/VerificationSuccess';
import ForgetPassword from
'./pages/auth/forget_password/ForgetPassword';
import Layout from './components/Common/Layout';
import RequireAuth from './components/Common/RequireAuth';
import NotFound from './pages/not_found/NotFound';

import 'react-loading-skeleton/dist/skeleton.css';
import "react-draft-wysiwyg/dist/react-draft-wysiwyg.css";
import { useState } from 'react';
// end auth

import Dashboard from './pages/dashboard/Dashboard';
import DashboardDetail from
'./pages/dashboard/detail/DashboardDetail';
import ReasonOfOutage from
'./pages/reason_of_outage/ReasonOfOutage';
import Report from './pages/report/Report';
import Profile from './pages/profile/Profile';
import BaseTransceiverStation from
'./pages/settings/base_transceiver_station/BaseTransceiverSt
ation';
import HistoryDashboard from
'./pages/history_dashboard/HistoryDashboard';
import RFODetailMass from
'./pages/reason_for_outage_trouble/detail_masal/RFODetailMas
s';
import RFODetailSingle from
'./pages/reason_of_outage/detail_mandiri/RFODetailSingle';
import DashboardRFOSingle from
'./pages/dashboard/rfo_single/DashboardRFOSingle';
import Statistics from './pages/statistics/Statistics';
import Users from './pages/settings/users/Users';
import Pop from './pages/settings/pop/Pop';
import SourceComplain from
'./pages/settings/source_complain/SourceComplain';
import Role from './pages/settings/role/Role';
import ReportCreate from
'./pages/report/create/ReportCreate';
import ReportDetail from
'./pages/report/detail/ReportDetail';
import Shift from './pages/settings/shift/Shift';
import ReasonForOutageTrouble from
'./pages/reason_for_outage_trouble/ReasonForOutageTrouble';
```

```
function App() {
  const [render, setRender] = useState(true);

  const getConditionRender = (event) => {
    setRender(event);
  };

  return (
    <div className="App">
      <Toaster />
      <Routes>
        <Route element={<Layout render={getConditionRender}
/>}>
          {/* public routes */}
          <Route index element={<SignIn />} />
          <Route path="/sign_in" element={render && <SignIn
/>} />
          <Route path="/sign_up" element={render && <SignUp
/>} />
          <Route
            path="/verification_email"
            element={render && <VerificationEmail />}
          />
          <Route
            path="/verification"
            element={render && <VerificationSuccess />}
          />
          <Route
            path="/forget_password"
            element={render && <ForgetPassword />}
          />
        </Route>
        <Route path="*" element={render && <NotFound />} />
        {/* protected routes */}
        <Route element={<RequireAuth />}>
          <Route index element={<Dashboard />} />
          <Route path="/dashboard" element={<Dashboard />}
/>
          <Route path="/dashboard/detail/:id"
element={<DashboardDetail />} />
          <Route
            path="/dashboard/rfo_single/:id"
            element={<DashboardRFOSingle />}
          />
```

```
          <Route path="/reason_of_outage"
element={<ReasonOfOutage />} />
          <Route path="/reason_of_outage_gangguan"
element={<ReasonForOutageTrouble />} />
          <Route
            path="/reason_of_outage/detail_masal/:id"
            element={<RFODetailMass />}
          />
          <Route
            path="/reason_of_outage/detail_single/:id"
            element={<RFODetailSingle />}
          />
          <Route path="/report" element={<Report />} />
          <Route path="/report/create"
element={<ReportCreate />} />
          <Route path="/report/detail/:id"
element={<ReportDetail />} />
          <Route path="/history_dashboard"
element={<HistoryDashboard />} />
          <Route
            path="/history_dashboard/detail/:id"
            element={<DashboardDetail />}
          />
          <Route
            path="/history_dashboard/rfo_single/:id"
            element={<DashboardRFOSingle />}
          />
          <Route
            path="/history_dashboard/rfo_masal/:id"
            element={<RFODetailMass />}
          />
          <Route path="/profile" element={<Profile />} />
          <Route
            path="/base_transceiver_station"
            element={<BaseTransceiverStation />}
          />
          <Route path="/statistics" element={<Statistics />}
/>

          {/* for admin */}
          <Route path="/users" element={<Users />} />
          <Route path="/pop" element={<Pop />} />
          <Route path="/source_complain"
element={<SourceComplain />} />
          <Route path="/role" element={<Role />} />
```

```
          <Route path="/shift" element={<Shift />} />
          {/* end admin */}
          <Route path="*" element={<NotFound />} />
        </Route>
      </Routes>
    </div>
  );
}

export default App;
```

Lampiran 5 Listing Program Komponen App.js

Komponen Dashboard untuk Keluhan

```
import { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { useAllComplainMutation } from
'../../store/features/complain/complainApiSlice';
import { selectAllComplain, setComplain } from
'../../store/features/complain/complainSlice';
import { useAllPOPMutation } from
'../../store/features/pop/popApiSlice';
import { setPOP } from '../../store/features/pop/popSlice';
import { updateBreadcrumb } from
'../../store/features/breadcrumb/breadcrumbSlice';
import { useAllSumberKeluhanMutation } from
'../../store/features/sumber_keluhan/sumberKeluhanApiSlice';
import { setSumberKeluhan } from
'../../store/features/sumber_keluhan/sumberKeluhanSlice';
import { selectCurrentUser } from
'../../store/features/auth/authSlice';
import { selectModalState, setModal } from
'../../store/features/modal/modalSlice';
import catchError from '../../services/catchError';
import ComplainModalForm from './modal/ComplainModalForm';
import RFOMasalModal from './modal/RFOMasalModal';
import ReopenModal from '../history_dashboard/ReopenModal';

import { DoShowRFOTrouble, DeleteModal,  DoDelete, DoUpdate,
LabelStatus, LoaderGetData, Search, Button, Modal,
SelectStatusComplain, SelectPOP, LabelStatusPOP, DoDetail,
DoShowRFOComplain, DoRollbackStatus, ProgressTime } from
'../../components/index';

const initColumns = [
```

```
  'No',
  'POP',
  'Pelanggan',
  'Kontak',
  'Keluhan',
  'Progress',
  'Waktu',
  'Status',
  'Aksi',
];

const Dashboard = () => {
  const [columns, setColumns] = useState(initColumns);
  const [statusData, setStatusData] = useState('open');
  const [detail, setDetail] = useState(null);
  const [showLoading, setShowLoading] = useState(true);
  const navigate = useNavigate();
  const [rows, setRows] = useState([]);
  const [allComplain] = useAllComplainMutation();
  const dispatch = useDispatch();
  const { data: user } = useSelector(selectCurrentUser);
  const stateModal = useSelector(selectModalState);
  const [search, setSearch] = useState('');
  const [dataPOP, setdataPOP] = useState([]);
  const [pop, setPOPLocal] = useState('all');
  const dataRow = useSelector(selectAllComplain);

  const openModal = (modal) => {
    let newState;
    if (modal === 'add complain') {
      newState = { ...stateModal, dashboard: {
...stateModal.dashboard, showAddModalComplain: true } };
    } else if (modal === 'update complain') {
      newState = { ...stateModal, dashboard: {
...stateModal.dashboard, showUpdateModalComplain: true } };
    } else if (modal === 'modal rfo masal') {
      newState = { ...stateModal, dashboard: {
...stateModal.dashboard, showRFOTroubleModal: true } };
    } else if (modal === 'delete complain') {
      newState = { ...stateModal, dashboard: {
...stateModal.dashboard, showDeleteModalComplain: true } };
    } else if (modal === 'revert complain') {
      newState = { ...stateModal, dashboard: {
...stateModal.dashboard, showRevertModalComplain: true } };
    }
```

```
    dispatch(setModal(newState));
    window.scrollTo(0, 0);
  }

  const getAllComplain = async () => {
    setShowLoading(true);
    try {
      const data = await allComplain().unwrap();
      if (data.status === 'success' || data.status ===
'Success') {
        let dataFix;
        if (user?.role_id === 2) {
          const dataFilter = data.data.filter((item) => {
            if (item.pop_id === user.pop_id && item.status
=== statusData) {
              return item;
            }
          });
          dataFix = dataFilter
          dispatch(setComplain({ data: dataFix }));
          setRows(dataFix);
        } else {
          const dataFilter = data.data.filter((item) => {
            if (item.status === statusData) {
              return item;
            }
          });
          dispatch(setComplain({ ...data }));
          setRows(dataFilter);
        }

        setTimeout(() => {
          setShowLoading(false);
        }, 1500);
      } else {
        setRows([]);
        setTimeout(() => {
          setShowLoading(false);
        }, 1500);
        catchError(data, false);
      }
    } catch (error) {
      setRows([]);
      setTimeout(() => {
        setShowLoading(false);
```

```
      }, 1500);
      catchError(error, false);
    }
  };

  const onHandleSearch = (event) => {
    event.preventDefault();
    setSearch(event.target.value);

    if (event.target.value.length > 0) {
      const regex = new RegExp(search, 'ig');
      const searchResult = dataRow.data.filter((item) => {
        if (item.status === statusData && +item.pop.id_pop
=== +pop) {
          if (item.id_pelanggan.match(regex) ||
item.nama_pelanggan.match(regex) ||
item.nama_pelapor.match(regex) ||
item.nomor_pelapor.match(regex)) {
            return item;
          }
        }
        if (item.status === statusData && pop === 'all') {
          if (item.id_pelanggan.match(regex) ||
item.nama_pelanggan.match(regex) ||
item.nama_pelapor.match(regex) ||
item.nomor_pelapor.match(regex)) {
            return item;
          }
        }
      });
      setRows(searchResult);
    } else {
      setRows(dataRow.data.filter((item) => {
        if (item.status === statusData && +item.pop.id_pop
=== +pop) {
          return item;
        }

        if (item.status === statusData && pop === 'all') {
          return item;
        }
      }));
    }
  }
```

```
  const handlePOP = (event) => {
    setPOPLocal(event.target.value);
    const dataChanged = dataRow.data.filter((item) => {
      if (+item.pop_id === +event.target.value &&
item.status === statusData) {
        return item;
      }
    })
    if (event.target.value === 'all') {
      setRows(dataRow.data.filter((item) => item.status ===
statusData));
    } else {
      setRows(dataChanged);
    }
  };

  const [allPOP] = useAllPOPMutation();

  const [allSumberKeluhan] = useAllSumberKeluhanMutation();

  const getAllSumberKeluhan = async () => {
    try {
      const data = await allSumberKeluhan().unwrap();
      if (data.status === 'success' || data.status ===
'Success') {
        dispatch(setSumberKeluhan({ ...data }));
      } else {
        catchError(data, true);
      }
    } catch (error) {
      catchError(error, true);
    }
  };

  const getAllPOP = async () => {
    try {
      const data = await allPOP().unwrap();
      if (data.status === 'success' || data.status ===
'Success') {
        let dataFix;
        if (user?.role_id === 2) {
          const dataFilter = data.data.filter((pop) => {
            if (pop.id_pop === user.pop_id) {
              return pop;
            }
```

```
          });
          dataFix = dataFilter
        } else {
          dataFix = data.data
        }
        dispatch(setPOP({ data: dataFix }));
        setdataPOP(dataFix);
      } else {
        catchError(data, true);
      }
    } catch (error) {
      catchError(error, true);
    }
  };

  useEffect(() => {
    dispatch(updateBreadcrumb([{ path: '/dashboard', title:
'Dasbor' }]))
    getAllPOP()
    getAllSumberKeluhan();
    getAllComplain();

    if (user?.role_id === 0) {
      setColumns([
        'No',
        'POP',
        'Pelanggan',
        'Kontak',
        'Keluhan',
        'Progress',
        'Waktu',
        'Status',
        'Sentimen',
        'Aksi',
      ]);
    }
    const intervalId = setInterval(() => {
      getAllComplain();
    }, 30000);
    return () => {
      clearInterval(intervalId);
    };
  }, []);

  const doCreateNotificationProgress = (data) => {
```

```
    if (data.notifikasi.length > 0) {
      let i = 0;
      data.notifikasi.forEach(item => {
        item.notifikasi_read.forEach(read => {
          if (+user.id_user === +read.user_id) {
            if (!read.is_read) {
              i++;
            }
          }
        })
      })

      if (i > 0) {
        return (
          <span className="badge bg-blue-700 border-none
text-white">
            + {i}
          </span>
        )
      }
    }
  }

  const handleStatus = (event) => {
    setStatusData(event.target.value);
    const dataChanged = dataRow.data.filter((item) => {
      if (item.status === event.target.value &&
+item.pop.id_pop === +pop) {
        return item;
      }
      if (item.status === event.target.value && pop ===
'all') {
        return item;
      }
    })
    setRows(dataChanged);
  };

  const getInfo = ($event) => {
    if ($event.status === 'success') {
      getAllComplain();
    }
  };

  const addData = () => {
```

```
    setDetail(null);
    openModal('add complain');
  }

  const editData = (item) => {
    setDetail(item);
    openModal('update complain');
  }

  const deleteData = (item) => {
    setDetail(item);
    openModal('delete complain');
  }

  const detailData = (item) => {
    navigate(`/dashboard/detail/${item.id_keluhan}`);
  }

  const RFOKeluhan = (item) => {
    navigate(`/dashboard/rfo_single/${item.id_keluhan}?id_rf
o=${item.rfo_keluhan_id}`);
  }

  const RFOMasal = (item) => {
    setDetail(item);
    openModal('modal rfo masal');
  }

  const RFOMasalDetail = (item) => {
    setDetail(item);
    navigate(
      `/reason_of_outage/detail_masal/${item.rfo_gangguan_id
}`
    );
  }

  const rollbackStatus = (item) => {
    setDetail(item);
    openModal('revert complain');
  }

  return (
    <div>
      {statusData === 'open' && <Button
onClick={addData}>Tambah</Button>}
```

```
      <div className="gap-5 mt-5 flex flex-col md:flex
md:flex-row">
        <div className="form-control w-full md:w-52">
          <SelectStatusComplain handleStatus={handleStatus}
all={false} />
        </div>

        <div className="form-control w-full md:w-52">
          <SelectPOP dataPOP={dataPOP} handlePOP={handlePOP}
/>
        </div>
        <div className="tooltip tooltip-right" data-
tip="Cari berdasarkan id_pelanggan, nama_pelanggan,
nama_pelapor, nomor_pelapor">
          <Search search={search}
onHandleSearch={onHandleSearch} placeholder={'Cari
berdasarkan id_pelanggan, nama_pelanggan, nama_pelapor,
nomor_pelapor'} />
        </div>
      </div>

      <div className="overflow-x-auto mt-8">
        <table className="table table-zebra w-full">
          <thead>
            <tr>
              {columns?.map((item, index) => (
                <th key={index} className="text-
center">{item}</th>
              ))}
            </tr>
          </thead>
          <tbody>
            {rows?.map((item, index) => (
              <tr key={index} className="text-center">
                <th className="bg-opacity-100">{index +
1}</th>
                <td className="text-left">
                  <LabelStatusPOP status={item?.pop?.pop} />
                </td>
                <td className="text-
left">{(item?.id_pelanggan)} - {(item?.nama_pelanggan)}</td>
                <td className="text-
left">{item?.nama_pelapor} - {item?.nomor_pelapor}</td>
                <td className="text-left">
```

```
                  <div dangerouslySetInnerHTML={{ __html:
item?.keluhan }} />
                </td>
                <td >
                  <div className="flex gap-2">
                    <div dangerouslySetInnerHTML={{ __html:
item?.balasan.length > 0 ? item?.balasan[item.balasan.length
- 1].balasan.slice(0, 100) : 'Belum ada tindakan' }}
className={`${item?.balasan.length === 0 ? 'text-center
font-semibold badge bg-red-500 border-0 text-white' : ''}`}
/>
                    <div>{user &&
doCreateNotificationProgress(item)}</div>
                  </div>
                </td>
                <td className="text-left">
                  <ProgressTime item={item} />
                </td>
                <td>
                  <LabelStatus status={item?.status} />
                </td>
                {user?.role_id === 0 &&
<td>{item?.sentimen_analisis || '-'}</td>}
                <td>
                  <div className="flex flex-row gap-3
justify-center">
                    {statusData === 'open' ? (
                      <>
                        <DoUpdate onClick={() =>
editData(item)} />
                        <DoDelete onClick={() =>
deleteData(item)} />
                        <DoDetail onClick={() =>
detailData(item)} />
                        <DoShowRFOComplain onClick={() =>
RFOKeluhan(item)} />
                        <DoShowRFOTrouble onClick={() =>
RFOMasal(item)} />
                      </>
                    ) : (
                      <>
                        <DoRollbackStatus onClick={() =>
rollbackStatus(item)} />
                        <DoDetail onClick={() =>
detailData(item)} />
```

```
                          {item.rfo_keluhan_id !== null &&
<DoShowRFOComplain onClick={() => RFOKeluhan(item)} />}
                          {item.rfo_gangguan_id !== null &&
<DoShowRFOTrouble onClick={() => RFOMasalDetail(item)} />}
                        </>
                      )}
                    </div>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
        {showLoading && <LoaderGetData />}
        <Modal>
          {stateModal?.dashboard?.showAddModalComplain &&
<ComplainModalForm stateModal={stateModal} detail={detail}
getInfo={getInfo} />}
          {stateModal?.dashboard?.showRFOTroubleModal &&
<RFOMasalModal stateModal={stateModal} detail={detail}
getInfo={getInfo} />}
          {stateModal?.dashboard?.showDeleteModalComplain &&
<DeleteModal stateModal={stateModal} detail={detail}
getInfo={getInfo} title="keluhan" />}
          {stateModal?.dashboard?.showUpdateModalComplain &&
<ComplainModalForm stateModal={stateModal} detail={detail}
getInfo={getInfo} />}
          {stateModal?.dashboard?.showRevertModalComplain &&
<ReopenModal stateModal={stateModal} detail={detail}
getInfo={getInfo} />}
        </Modal>
      </div>
    );
}

export default Dashboard;
```

Lampiran 6 Listing Program Komponen Keluhan

Komponen Balasan atau Detail Keluhan

```
import { useState, useEffect } from 'react';
import { useDispatch, useSelector } from 'react-redux';
import { useLocation, useNavigate, useParams } from 'react-
router-dom';
import { Formik, Form } from 'formik';
```

```
import { useAddReplyMutation, useComplainByIdMutation,
useLampiranFileBalasanMutation } from
'../../../store/features/complain/complainApiSlice';
import { setComplainById } from
'../../../store/features/complain/complainSlice';
import { selectBreadcrumb, updateBreadcrumb } from
'../../../store/features/breadcrumb/breadcrumbSlice';
import { selectCurrentUser } from
'../../../store/features/auth/authSlice';
import { ReplySchema } from
'../../../utils/schema_validation_form';
import { usePostNotificationMutation,
useReadAllNotificationComplainMutation,
useStoreAllNotificationMutation } from
'../../../store/features/notification/notificationApiSlice';
import catchError from '../../../services/catchError';
import handleResponse from
'../../../services/handleResponse';
import { Button, Progress, FirstComplain,
SectionInformation, UploadFile, TextEditor } from
'../../../components';

function DashboardDetail({ rfoSingle, idComplain }) {
  const location = useLocation();
  const [detailComplain, setDetailComplain] =
useState(null);
  const [resetTextEditor, setResetTextEditor] =
useState(false);
  const [filesLocal, setFilesLocal] = useState([]);
  const { id } = useParams();
  const navigate = useNavigate();
  const [complainById] = useComplainByIdMutation();
  const [addReply] = useAddReplyMutation();

  const dispatch = useDispatch();
  const [lampiranFileBalasan] =
useLampiranFileBalasanMutation();
  const [storeAllNotification] =
useStoreAllNotificationMutation();
  let [countRequest, setCountRequest] = useState(0);
  const [resetFile, setResetFile] = useState(false);

  const navigasi = useSelector(selectBreadcrumb);
  const onHandleFileUpload = ($event) => {
    setFilesLocal($event);
```

```
  }

  const getComplainById = async () => {
    try {
      let idConditional;
      if (idComplain !== undefined) {
        idConditional = idComplain;
      } else {
        idConditional = id;
      }
      const data = await
complainById(idConditional).unwrap();
      dispatch(setComplainById({ ...data }));
      setDetailComplain(data.data);
      if (countRequest === 0) {
        if
(window.location.href.includes('/dashboard/detail/')) {
          doReadAllNotification(data.data);
        }
      }
      setCountRequest(countRequest++);
    } catch (error) {
      catchError(error, true)
    }
  };

  const { data: user } = useSelector(selectCurrentUser);
  const historyUrl = location.pathname.includes('history');
  const [postNotification] = usePostNotificationMutation();

  const [readAllNotificationComplain] =
useReadAllNotificationComplainMutation()

  const doReadAllNotification = async (data) => {
    const body = {
      keluhan_id: data.id_keluhan
    };
    try {
      await readAllNotificationComplain({ body }).unwrap();
      // getAllNotification();
    } catch (error) {
      catchError(error, false);
      throw new Error("Unsuccess read all notification");
    }
  }
```

```
  useEffect(() => {
    getComplainById();
    if (idComplain === undefined) {
      const data = [...navigasi.data, { path:
`/dashboard/detail/${id}`, title: 'Detail Dasbor' }]
      dispatch(updateBreadcrumb(data))

      setTimeout(() => {
        window.scrollTo(0, document.body.scrollHeight);
      }, 500);

      const intervalId = setInterval(() => {
        getComplainById();
      }, 30000);
      return () => {
        clearInterval(intervalId);
      };
    }
  }, [])

  const doPostNotification = async (keluhan_id) => {
    try {
      const body = {
        keluhan_id,
        pop_id: user.pop_id,
        id_response: 1
      }
      const data = await postNotification({ body
}).unwrap();
      return data;
    } catch (error) {
      catchError(error, true);
    }
  }

  const doStoreAllNotiification = async (notifikasi_id) => {
    try {
      const body = {
        notifikasi_id,
      }
      const data = await storeAllNotification({ body
}).unwrap();
      return data;
    } catch (error) {
```

```
      catchError(error, true);
    }
  }

  const onSubmitData = async (payload, resetForm) => {
    try {
      const body = {
        balasan: payload.balasan,
        user_id: user.id_user,
        keluhan_id: detailComplain.id_keluhan
      }
      const add = await addReply({ ...body });
      if (add.data.status === 'success' || add.data.status
=== 'Success') {
        resetForm()
        setResetTextEditor(true);
        handleResponse(add);
        if (filesLocal.length > 0) {
          const formData = new FormData();
          formData.append('balasan_id',
add?.data?.id_balasan);
          for (let index = 0; index < filesLocal.length;
index++) {
            formData.append(`path[${index}]`,
filesLocal[index])
          }

          await lampiranFileBalasan({ body: formData
}).unwrap();
        }
        const dataNotification = await
doPostNotification(detailComplain.id_keluhan);
        if (dataNotification?.status === 'Success' ||
dataNotification?.status === 'success') {
          const dataPost = await
doStoreAllNotiification(dataNotification?.notifikasi?.id_not
ifikasi);
          if (dataPost?.status === 'Success' ||
dataPost?.status === 'success') {
            getComplainById();
            setResetTextEditor(false);
            setFilesLocal([]);
            setResetFile(true);
            setTimeout(() => {
              setResetFile(false);
```

```
            }, 500);
          } else {
            catchError(dataPost, true);
            setTimeout(() => {
              setResetFile(false);
            }, 500);
          }
        } else {
          setResetTextEditor(false);
          catchError(dataNotification, true);
        }
      } else {
        catchError(add, true);
      }
    } catch (error) {
      catchError(error, true);
    }
  }

  return (
    <div>
      {!rfoSingle && <SectionInformation
detailComplain={detailComplain} />}

      <div className="flex w-full flex-col py-5">
        <FirstComplain detailComplain={detailComplain} />
        <hr />

        {
          detailComplain?.balasan?.length > 0 &&
detailComplain?.balasan?.map((item, index) => (
            <Progress item={item} key={index} />
          ))
        }

        {!historyUrl && !rfoSingle && detailComplain?.status
=== 'open'
          ? (
            <Formik
              enableReinitialize
              validationSchema={ReplySchema}
              initialValues={{ balasan: '' }}
              onSubmit={(values, { resetForm }) => {
                onSubmitData(values, resetForm);
              }}
```

```
            >
              {({
                values,
                errors,
                touched,
                isValid,
                setFieldValue
              }) => (
                <Form id='answer'>
                  <div className="form-control">
                    <label htmlFor="balasan"
className="label">
                      <span className="label-text">
Balasan</span>
                    </label>
                    <TextEditor
                      setFieldValue={(val) =>
setFieldValue('balasan', val)}
                      value={values.balasan}
                      resetData={resetTextEditor}
                    />
                    {errors.balasan && touched.balasan ? (
                      <div className="label label-text text-
red-500">{errors.balasan}</div>
                    ) : null}
                  </div>

                  <UploadFile
getFile={onHandleFileUpload}  reset={resetFile} />
                  <div className="text-center items-center
justify-center mt-10">
                    <Button className={`mr-5`} onClick={()
=> navigate('/dashboard')}>Kembali</Button>
                    <Button type={"submit"}
disabled={!isValid} className={`btn-
success`}>Simpan</Button>
                  </div>
                </Form>
              )}
            </Formik>
          ) : !rfoSingle && (
            <div className="text-center items-center
justify-center mt-10">
```

```
              <Button className={`mr-5`} onClick={() =>
historyUrl ? navigate('/history_dashboard') :
navigate('/dashboard')}>Kembali</Button>
            </div>
          )}
      </div>
    </div>
  );
}

export default DashboardDetail;
```

Lampiran 7 Listing Program Komponen Balasan

Konfigurasi Pusher JS

```
import * as PusherPushNotifications from '@pusher/push-
notifications-web';
import { REACT_APP_PUSHER_APP_INSTANCEID_BEAM } from
'../config';
import catchError from '../services/catchError';

function usubscribeBeamAndLogout() {
    const beamsClient = new PusherPushNotifications.Client({
        instanceId: REACT_APP_PUSHER_APP_INSTANCEID_BEAM,
        });

        beamsClient.stop()
        .then(() => console.log('Beams SDK has been
stopped'))
        .catch(e => console.error('Could not stop Beams
SDK', e));

        beamsClient.clearAllState()
        .then(() => console.log('Beams state has been
cleared'))
        .catch(e => console.error('Could not clear Beams
state', e));

      localStorage.clear();
}

function startBeamClient() {
  const beamsClient = new PusherPushNotifications.Client({
    instanceId: REACT_APP_PUSHER_APP_INSTANCEID_BEAM,
    });
```

```
    beamsClient.start()
    .then((beamsClient) => beamsClient.getDeviceId())
    .then((deviceId) => deviceId)
    .then(() => beamsClient.addDeviceInterest("update"))
    .then(() => beamsClient.getDeviceInterests())
    .then((interests) => interests).catch(err =>
console.log(err));
}

export {usubscribeBeamAndLogout, startBeamClient};
```

Lampiran 8 Listing Program Konfigurasi Pusher JS

Konfigurasi State Management Redux Toolkit

```
import { configureStore } from '@reduxjs/toolkit';
import { apiSlice } from '../api/apiSlice';
import authReducer from './features/auth/authSlice';
import btsReducer from './features/bts/btsSlice';
import complainReducer from
'./features/complain/complainSlice';
import complainHistoryReducer from
'./features/complain_history/complainHistorySlice';
import rfoReducer from './features/rfo/rfoSlice';
import popReducer from './features/pop/popSlice';
import teamReducer from './features/team/teamSlice';
import sumberKeluhanReducer from
'./features/sumber_keluhan/sumberKeluhanSlice';
import breadcrumbReducer from
'./features/breadcrumb/breadcrumbSlice';
import usersReducer from './features/users/usersSlice';
import reportReducer from './features/report/reportSlice';
import shiftReducer from './features/shift/shiftSlice';
import notificationReducer from
'./features/notification/notificationSlice';
import modalReducer from './features/modal/modalSlice';
// eslint-disable-next-line import/prefer-default-export
export const store = configureStore({
  reducer: {
    [apiSlice.reducerPath]: apiSlice.reducer,
    auth: authReducer,
    breadcrumb: breadcrumbReducer,
    complain: complainReducer,
    rfo: rfoReducer,
```

```
    complain_history: complainHistoryReducer,
    pop: popReducer,
    team: teamReducer,
    sumber_keluhan: sumberKeluhanReducer,
    bts: btsReducer,
    users: usersReducer,
    report: reportReducer,
    shift: shiftReducer,
    notification: notificationReducer,
    modal: modalReducer,
  },
  middleware: (getDefaultMiddleware) =>
getDefaultMiddleware().concat(apiSlice.middleware),
  devTools: process.env.NODE_ENV !== 'production',
});
```

Lampiran 9 Listing Program Konfigurasi State Management Redux

Konfigurasi Middleware JWT Token Pada Header di FrontEnd

```
import { createApi, fetchBaseQuery } from
'@reduxjs/toolkit/query/react';
import { setLogOut } from
'../store/features/auth/authSlice';
import { clearBTS } from '../store/features/bts/btsSlice';
import { clearComplain } from
'../store/features/complain/complainSlice';
import catchError from '../services/catchError';
import { clearComplainHistory } from
'../store/features/complain_history/complainHistorySlice';
import { clearModal } from
'../store/features/modal/modalSlice';
import { clearNotification } from
'../store/features/notification/notificationSlice';
import { clearPOP } from '../store/features/pop/popSlice';
import { clearReport } from
'../store/features/report/reportSlice';
import { clearRFO } from '../store/features/rfo/rfoSlice';
import { clearShift } from
'../store/features/shift/shiftSlice';
import { clearSumberKeluhan } from
'../store/features/sumber_keluhan/sumberKeluhanSlice';
```

```
import { clearTeam } from
'../store/features/team/teamSlice';
import { clearUsers } from
'../store/features/users/usersSlice';

const baseQuery = fetchBaseQuery({
  baseUrl: 'http://localhost:8000/api',
  mode: 'cors',
  prepareHeaders: (headers, { getState, endpoint }) => {
    const token = getState().auth.data;
    // console.log(endpoint, 'endpoint');
    if (token?.bearer_token) {
      headers.set('authorization', `Bearer
${token?.bearer_token}`);
    }
    return headers;
  },
});

const baseQueryWithReauth = async (args, api, extraOptions)
=> {
  try {
    const result = await baseQuery(args, api, extraOptions);
    if (result?.error) {
      const data = {
        status : result?.error.status ||
result?.error?.originalStatus,
        data: {
          message: result?.error?.data?.message ||
result?.error?.status.toString()
        }
      }
      if (data?.data?.message?.includes('Keluhan') ||
data?.data?.message?.includes('Statistic')) {
        catchError(data, false);
      } else {
        catchError(data, true);
      }
    }
    if (result?.error?.data?.message?.match(/Please login
first/ig) || result?.error?.data?.message?.match(/Please
activate account/ig)) {
      api.dispatch((action) => {
        action(setLogOut());
        action(clearBTS());
```

```
        action(clearComplain());
        action(clearComplainHistory());
        action(clearModal());
        action(clearNotification());
        action(clearPOP());
        action(clearReport());
        action(clearRFO());
        action(clearShift());
        action(clearSumberKeluhan());
        action(clearTeam());
        action(clearUsers());
      });
      localStorage.clear();
      window.location.reload();
    }
    return result;
  } catch (error) {
    catchError(error, true);
  }
};

export const apiSlice = createApi({
  baseQuery: baseQueryWithReauth,
  endpoints: (builder) => ({}),
});
```

Lampiran 10 Listing Program JWT Token Header

Konfigurasi Endpoint Keluhan Pada Redux Toolkit Js

```
import { apiSlice } from '../../../api/apiSlice';

export const complainApiSlice = apiSlice.injectEndpoints({
  endpoints: (builder) => ({
    allComplain: builder.mutation({
      query: () => ({
        url: '/keluhan',
        method: 'GET',
      }),
    }),
    complainById: builder.mutation({
      query: (id) => ({
```

```
      url: `/keluhan/${id}`,
      method: 'GET',
    }),
  }),
  deleteComplain: builder.mutation({
    query: (id) => ({
      url: `/delete-keluhan/${id}`,
      method: 'GET',
    }),
  }),
  addComplain: builder.mutation({
    query: (body) => ({
      url: `/keluhan`,
      method: 'POST',
      body
    }),
  }),
  updateComplain: builder.mutation({
    query: ({ id, body }) => ({
      url: `/keluhan/${id}`,
      method: 'PUT',
      body: { ...body },
    }),
  }),
  addReply: builder.mutation({
    query: (body) => ({
      url: `/balasan`,
      method: 'POST',
      body,
    }),
  }),
  complainClosed: builder.mutation({
    query: (id) => ({
      url: `/close/${id}`,
      method: 'PUT',
    }),
  }),
  lampiranFile: builder.mutation({
    query: ({ body }) => ({
      url: `/lampiran-keluhan`,
      method: 'POST',
      body
    }),
  }),
  lampiranFileBalasan: builder.mutation({
```

```
      query: ({ body }) => ({
        url: `/lampiran-balasan`,
        method: 'POST',
        body
      }),
    }),
  }),
});

export const { useAllComplainMutation,
useComplainByIdMutation, useDeleteComplainMutation,
useAddComplainMutation, useUpdateComplainMutation,
useAddReplyMutation, useComplainClosedMutation,
useLampiranFileMutation, useLampiranFileBalasanMutation } =
complainApiSlice;
```

Lampiran 11 Listing Program Endpoint Keluhan